Data Graph Formulation as the Minimum-Weight Maximum-Entropy Problem

Samuel de Sousa and Walter G. Kropatsch

Vienna University of Technology
Pattern Recognition and Image Processing Group
Vienna, Austria
sam,krw@prip.tuwien.ac.at

Abstract. Consider a point-set coming from an object which was sampled using a digital sensor (depth range, camera, etc). We are interested in finding a graph that would represent that point-set according to some properties. Such a representation would allow us to match two objects (graphs) by exploiting topological properties instead of solely relying on geometrical properties. The Delaunay triangulation is a common out off-the-shelf strategy to triangulate a point-set and it is used by many researchers as the standard way to create the so called data-graph and despite its positive properties, there are also some drawbacks. We are interested in generating a graph with the following properties: the graph is (i) as unique as possible, (ii) and as discriminative as possible regarding the degree distribution. We pose a combinatorial optimization problem (Min-Weight Max-Entropy Problem) to build such a graph by minimizing the total weight cost of the edges and at the same time maximizing the entropy of the degree distribution. Our optimization approach is based on Dynamic Programming (DP) and yields a polynomial time algorithm.

Keywords: data-graph, graph realization, combinatorial optimization

1 Introduction

In many applications it is necessary to create a graph out of an unstructured point-set. Such a point-set could represent the projection of an object onto an image or locations of key features. A common problem in Computer Vision consists of the registration of two or more point-sets. As a result, one would (i) obtain the transformation that maps one set towards the other and (ii) find the pairwise correspondence between points in all sets.

A graph created out of a point-set will be referred here as the data-graph. A common procedure consists of (i) creating the data-graph using the Delaunay triangulation and (ii) performing the registration using an optimization procedure [1, 2, 11]. The Delaunay triangulation [3] is based on the condition that no other point should lie inside the circumcircle of any triangle. It is not clear, though, if the Delaunay triangulation is always the best solution for all possible tasks in Computer Vision when a data-graph is required. There are several

alternative methods such as Reeb graphs [13], Gabriel Graph [6], and also the Euclidean Minimum Spanning Tree (EMST). Many researchers propose methods of data-graph construction focusing either on aesthetic aspects [12] or designing their own criteria, such as the fan-shaped triangulation of Lian and Zhang [10].

Developing a more unique representation of a graph is not a new concept and an relevant paper in the topic was produced by Dickinson et al. [4] where they focus on a class of graphs which have unique representation of the node labels, a representation ρ of graph $g = (V, E, \alpha, \beta)$ is created where α and β are functions assignings labels to the vertices and edges respectively. They can find if two graphs g' and g'' are isomorphics by comparing their representations $\rho(g')$ and $\rho(g'')$ in $O(N^2)$. Our work differs in nature with the previous work since we are not only interested in an isomorphism between two graphs, since many isomorphisms can still be possible and the points couldn't be uniquely identified due to the many possibles solutions. We aim at creating a graph in which the number of possible isomorphims would be as small as possible. We pose this problem as an optimization problem where we call it the Minimum-Weight Maximum-Entropy Problem which captures the desired behaviour a graph should be designed and we provide an efficient polynomial time algorithm based on Dynamic Programming (DP) to solve it. To the best of our knowledge, we are the first ones to tackle the matching problem in this fashion.

The remainder of this paper is organized as follows: Section 2 formulates the problem being addressed in this paper and the difficulties associated with the minimization of its cost function. Section 3 introduces the *Near Homogeneous Degree Distribution (NHDD)* property which is used for building up the solution. As our solution possesses a recurrence relation, we design a Dynamic Programming algorithm on Section 4. Finally, we draw our conclusions and future work on Section 5.

2 The Minimum-Weight Maximum-Entropy Problem

As the goal of our paper is to design data-graphs which would ease the registration process, the first question to be asked is how we can define such metric, or which property an ideal data graph would have in order to make the registration process as trivial as possible. If we examine a regular graph, i.e. a graph in which all nodes have the same degree, we would notice that all nodes could be matched against all the other nodes, and the registration task would generate many ambiguous solutions. Therefore, if one succeeds to build a graph which is exactly opposite of a regular graph, i.e. a graph whose degree distribution is as diverse as possible, the registration process would be, then, alleviated.

In order to measure the diversity found in the degree distribution, we can calculate the Shannon entropy (H) for a graph $G(V, E)$ as follows:

$$\mathrm{H}(G) = -\sum_{v \in V_{\neq}} p(v) \log_2(p(v)). \tag{1}$$

where $p(v)$ is the probability of finding a node with a degree of v among all distinct degree values ($V_{\neq}$ is the set of distinct degree values of V). The entropy measures the uncertainty associated with a random variable. As defined in Eq. 1, a high entropy $\mathrm{H}(G)$ of a graph $G(V, E)$ would indicate a high "variability" in the distribution of nodes V. The converse is also true, a low entropy means low variability, as in a k-regular graph whose probability $p(k) = 1$ and $log(1) = 0$.

We aim at obtaining a graph with the highest entropy that would let us match the nodes more easily. Nevertheless, even if we are able to obtain a graph whose entropy is as high as possible, there is still the problem of ambiguity. There are multiple solutions with the same entropy value. Therefore, the second question we pose is how to generate a graph as unique as possible. Such question is important due to the fact that it would allow us to match two graphs based only on their degree values. We would like to uniquely identify the nodes unless all points in the graph are equidistant, in this case many possible solutions still exist. To achieve that, we decided to minimize the total weighted edge cost of our graph. We call this problem *the Minimum-Weight Maximum-Entropy* (MWME):

Definition 1. *The Minimum-Weight Maximum-Entropy (MWME) is the problem of estimating an edge-induced subgraph of a graph whose entropy of the node degrees is maximum and the total edge weight is minimum.*

Given a point set P, we create a complete graph $K_{|P|}$ using a metric[1] function as the edge weights. Let W denote the weighted edges of $K_{|P|}$. We define a binary vector U that induces an edge-subgraph $G[U]$ composed of all nodes of $K_{|P|}$ and edges $\{W_i | U_i = 1\}$. We search for the vector U which minimizes the cost:

$$\begin{aligned} \underset{U}{\text{minimize}} \quad & \sum_{i=1}^{|W|} W_i U_i, \\ \text{subject to} \quad & \mathrm{H}(F) \leq \mathrm{H}(G[U]), \forall F \\ & U \in \{0,1\}^{|W|}. \end{aligned} \tag{2}$$

under the constraint that the entropy $\mathrm{H}(G[U])$ of the induced subgraph $G[U]$ is maximum, i.e. for any graph F, the entropy $\mathrm{H}(F)$ will be lower or equal to our edge induced subgraph $G[U]$ The second constraint states that the optimization variable is discrete, we either add the edge W_i to our graph $G[U]$ when $U_i = 1$ or we do not add such an edge ($U_i = 0$).

We propose a dynamic programming algorithm that minimizes Eq. 2 by looking deeper into some properties of the desired induced graph $G[U]$. It is important to mention that we cannot guarantee unique solutions. The reason for that can be visualized in Fig. 1(a). By constructing a graph out of a regular polygon, we could rotate all nodes and the total edge cost would remain the same as well as the entropy. Therefore, there would be many possible solutions.

[1] e.g. the Euclidean distance

3 The Near Homogeneous Degree Distribution (NHDD)

In order to minimize the cost function provided in Eq. 2, we will decompose the problem first in the entropy maximization and then in the edge cost minimization. In this section we explain how we guarantee the maximum possible entropy in a graph and later how to incorporate our graph with the minimum edge cost.

We need to know, theoretically, how many nodes a simple graph[2] can have with distinct degree values in order to create a graph with the highest variability. This problem is closely related to the graph realization problem [5] which consists of determining if such a sequence of degrees can be feasible for a simple graph (called graphic sequence). This problem has been addressed by Erdős-Gallai [5] and Havel-Hakimi [7, 8]. We would like to obtain the maximal variability in a graph, our problem could be considered as generating the graphic sequence with highest variability. Theorem 1 states the maximum variability in a graph.

Theorem 1. *For every simple graph $G(V, E)$ with $|V| > 2$, there are two nodes with the same degree.*

Proof (Kocay and Kreher [9]). A simple graph does not allow parallel edges and self-loops, therefore, the highest degree is $|V| - 1$. If we create a degree sequence in which all nodes have different degree values (the highest possible variability), this would mean that the nodes would have to be $V = (0, 1, \ldots, |V| - 1)$. If one node has a degree of $|V| - 1$, it means it is connected to all the other nodes, but the degree zero means that one node is not connected to any other. Thus, degree values of 0 and $|V| - 1$ are mutually exclusive and cannot coexist in the same graph, since this leaves only $|V| - 1$ values for $|V|$ nodes, by the pidgeon hole principle, at least two nodes must have the same degree.

Since it is not feasible to create a graph whose nodes have all distinctive degree values, the highest possible variability is $|V| - 1$ (Def. 2).

Definition 2. *A graph $G(V, E)$ fulfills the Near Homogeneous Degree Distribution (NHDD) property when it contains $|V| - 1$ nodes with different degrees.*

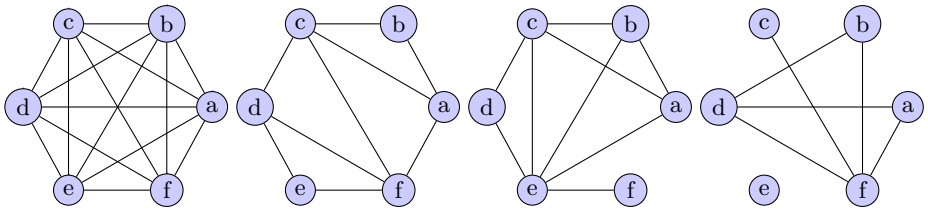
An induced subgraph for our optimization function (Eq. 2) is only feasible if it fulfills the NHDD. In Theorem 4, we prove that such a graph has maximum entropy. For now, we show that there are two of feasible graphs (Def. 3).

Definition 3. *G_N is a connected graph which fulfills the NHDD property and G_N° is a graph which fulfills the NHDD but it contains one isolated node.*

Lemma 1 shows how we can generate the two possible graphs (G_N and G_N°) fulfilling the NHDD definition.

Lemma 1. *For any integer $N \geq 3$, it is possible to generate both G_N and G_N°.*

[2] A simple graph is a graph which does not contain parallel edges and self-loops.



$H(K_6) = 0$ (a) Complete K_6 $H(L_6) = 0.91$ (b) Delaunay L_6 $H(G_6) = 2.25$ (c) NHDD G_6 $H(G_6^\circ) = 2.25$ (d) NHDD G_6°

Fig. 1: Some realizations of a six vertices graph along with their entropy. Graph (a) is a complete graph, Graph (b) is a triangulation and Graphs (c) and (d) fulfill the NHDD property with the highest entropy values.

Proof. We start with a base case of a graph G_N° with $N = 3$, whose degree distribution (the degree values of all nodes in the graph) is $D_3^\circ = (0, 1, 1)$. The NHDD is already fulfilled. Our inductive step states that this holds for any number $N \geq 3$. In order to build a solution for G_{N+1}, there are two possibilities:

a) G_N° has an isolated node. Then, by adding a node and connecting it to all the others, we will obtain a G_{N+1} since the degree of all the other nodes will be increased by one and the graph will be connected. $G_{N+1}^\circ = K_{N+1} - G_{N+1}$.

b) G_N does not have an isolated node. Therefore, we add a new isolated node and we obtain G_{N+1}° in which the NHDD is fulfilled. $G_{N+1} = K_{N+1} - G_{N+1}^\circ$ □

In the proof, we used the complement of a graph. In Lemma 2, we show that the complement of a graph fulfilling NHDD also fulfills the same property. Fig. 1 shows four different induced graphs with 6 nodes and their respective entropy values (Eq. 1). Graph (a) is a complete graph K_6, whose entropy is equal to zero. Graph (b) is created using a Delaunay triangulation (L_6) and its entropy is equal to 0.91. Graphs (c) and (d) are a G_6 and a G_6° respectively. It is clear that both G_N° and G_N are more distinctive than L_6 and K_6 and their entropy is significantly higher. The existence of an isolated node did not affect the entropy as the variability in the number of existing nodes is the same, however, as the nodes have the same distance, those solutions are not unique.

Lemma 2. *The complement graph of G_N also fulfills the NHDD property.*

Proof. The complement graph $\overline{G_N}$ is obtained by taking the difference of $K_N - G_N$. A G_N has distribution equal to $D_N = (1, \ldots, |V| - 1)$. A complete graph K_N has all nodes with degree equal to $|V| - 1$. Therefore, the degree distribution of $K_N - G_N$ will be $D_N^\circ = (|V| - 2, \ldots, 0)$ with one isolated node and maximum degree of $|V| - 2$.

Theorem 2. *For any integer number N, it is always possible to obtain a connected graph that fulfills the NHDD property.*

Proof. This proof comes out naturally as a consequence of Lemmas 1 and 2. For any integer N, we will obtain either G_N or G_N°. In case we obtain G_N°, we take the complement and we can always obtain a connected graph which is *NHDD.*

□

Since we have been speaking about the distinctive node degrees which would allow a one-to-one match between graphs, we can also determine which node would be the duplicated one (Lemma 3).

Lemma 3. *The repeated node for a G_N has degree equal to $\lfloor \frac{N}{2} \rfloor$.*

Proof. The proof will be by induction. We start with a base case of even parity with $N = 4$. Hence, the node degree distribution is $D_4 = (1, 2, 2, 3)$ and $v_4^{\ddagger} = 2$. Our induction step is then:

$$D_N = \left(1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor, \ldots, N-1\right). \tag{3}$$

By adding one isolated node as described in Lemma 1, we obtain:

$$D_{N+1}^{\circ} = \left(0, 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{2} \right\rfloor, \ldots, N-1\right) \tag{4}$$

The connected graph can be achieved by taking the complement of $\overline{D_{N+1}^{\circ}}$:

$$D_{N+1} = \left(N, \ldots, N - \left\lfloor \frac{N}{2} \right\rfloor, N - \left\lfloor \frac{N}{2} \right\rfloor, \ldots, 1\right) \tag{5}$$

We know that $N = \lfloor \frac{N}{2} \rfloor + \lfloor \frac{N+1}{2} \rfloor$, therefore, by plugging it back at Eq. 5, we obtain:

$$D_{N+1} = \left(1, \ldots, \left\lfloor \frac{N+1}{2} \right\rfloor, \left\lfloor \frac{N+1}{2} \right\rfloor, \ldots, N\right) \tag{6}$$

□

The number of edges of G_N and G_N° can be directly calculated (Theorem 3).

Theorem 3. *The number of edges of graphs G_N and G_N° is equal to $|E_N| = \frac{1}{2}\left(\frac{N(N-1)}{2} + \lfloor \frac{N}{2} \rfloor\right)$ and $|E_N^{\circ}| = \frac{1}{2}\left(\frac{(N-2)(N-1)}{2} + \lfloor \frac{N-1}{2} \rfloor\right)$.*

Proof. We will first prove for graph G_N. As proved in Lemma 3, the sum of degrees in a G_N is equal to:

$$\sum_{v \in G_N} deg(v) = 1 + \ldots + \left\lfloor \frac{N}{2} \right\rfloor + \left\lfloor \frac{N}{2} \right\rfloor + \ldots + N - 1 \tag{7}$$

which is equivalent to:

$$\sum_{v \in G_N} deg(v) = \sum_{i=1}^{N-1} i + \left\lfloor \frac{N}{2} \right\rfloor = \frac{N(N-1)}{2} + \left\lfloor \frac{N}{2} \right\rfloor \tag{8}$$

The handshake lemma states that $|E_N| = \sum deg(v)/2$, we arrive that the number of edges is:

$$|E_N| = \frac{1}{2}\left(\frac{N(N-1)}{2} + \left\lfloor \frac{N}{2} \right\rfloor\right) \tag{9}$$

By rewriting Equation 4, we conclude that:

$$|E_N^\circ| = \sum_{v \in G_N^\circ} deg(v) = \sum_{i=0}^{N-2} i + \left\lfloor \frac{N-1}{2} \right\rfloor = \frac{1}{2}\left(\frac{(N-2)(N-1)}{2} + \left\lfloor \frac{N-1}{2} \right\rfloor \right) \tag{10}$$

□

4 Optimization

The discussion about the NHDD property was pursued during the attempt to minimize our objective function (Eq. 2). If we would like to solve our optimization using that property, we need to prove that the graph we generate has the maximum possible entropy (Theorem 4):

Theorem 4. *The G_N has the maximum entropy for any graph with N nodes.*

Proof. The proof can be performed using the method of Lagrangian multipliers. The objective function to be maximized is defined in Eq. 1. There is one equality constraint as all probabilities should add up to one: $\sum p(v) = 1$. The Lagrangian takes the form of:

$$\mathcal{L}(H, \lambda) = -\sum_{v \in G_N} p(v) log(p(v)) + \lambda \left(\sum_{v \in G_N} p(v) - 1 \right). \tag{11}$$

In order to find the critical points, one needs to take the partial derivatives and set them equal to zero:

$$\frac{\partial \mathcal{L}}{\partial p(v)} = -log(p(v)) - 1 + \lambda = 0, \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \sum p(v) - 1 = 0. \tag{13}$$

Finally, considering that all $p(v)$ are the same and only dependent on λ:

$$p(v) = e^{\lambda - 1} = \frac{1}{N} \tag{14}$$

The uniform distribution is the solution for the maximum entropy. Therefore, there should be N distinctive values in the set of node degrees to have the maximal possible entropy. As proved in the Theorem 1, it is not possible for a simple graph to have an uniform distribution. Our NHDD has $N-1$ distinctive node degrees and it has the maximal entropy a graph could have. □

Theorem 5 calculates the entropy of graphs G_N or G_N° analytically.

Theorem 5. *The entropy of $H(G_N) = H(G_N^\circ) = \log_2 N - \frac{2}{N}$.*

Proof. In a probabilistic interpretation of the entropy, the $p(x)$ is the probability of a event x to happen. Our event is the occurrence of a degree x in G_N (or G_N°). There are $N-2$ events with probability $p(x) = {}^1/_N$ and one whose probability is $p(x) = {}^2/_N$. Therefore, the entropy of $H(G_N)$ and $H(G_N^\circ)$ is:

$$H(G_N) = \frac{N\log N}{N} + \frac{2\log N}{N} - \frac{2\log N}{N} - \frac{2}{N} = \log N - \frac{2}{N} \tag{15}$$

□

Data: Complete graph $K_N(P, W)$, $N \geq 3$;
1 **begin**
2 $dp \leftarrow Matrix(N, N, N, value = \infty)$;
3 // We create N solutions starting with point i (layer).
4 **for** $i \leftarrow 1$ **to** N **do**
5 $dp(i, 1, i : N) = 0; join = 0$;
6 // k is the capacity of the graph G_N (row).
7 **for** $k \leftarrow 2$ **to** N **do**
8 // j is the node to be added into G_N (column).
9 **for** $j \leftarrow 1$ **to** N **do**
10 **if** *join* **then**
11 $c_{new} \leftarrow dp(i, k-1, N) + \sum_{g \in G}^{|G|} W(j, g)$;
12 **else**
13 $c_{new} \leftarrow dp(i, k-1, N) + W(j, v_{k-1})$;
14 $dp(i, k, j) \leftarrow \min(dp(i, k, j-1), c_{new})$;
15 $join = \neg join$;
16 // The new solution i is at least as good as $i-1$.
17 $dp(i, n, n) \leftarrow \min(dp(i, n, n), dp(i-1, n, n))$;
18 **return** $TraceBack(dp)$.

Algorithm 1: The `n1graph` algorithm which generates a G_N graph.

Our `n1graph` algorithm (Alg. 1) starts by defining a Matrix dp (line 2) of dimensions N^3 whose elements are ∞. This matrix will hold the cost to be minimized. We produce N solutions iteratively (line 4), in which the cost of solution i will be most as low as $i-1$ (line 17). Every iteration of i could be visualized as one layer of dp and the initialization occurs by setting the costs from i to N equal to zero (line 5). Since the weights lie in the edges, the initialization indicates which node is chosen at the moment (i) and it is zero since there are no edges in G_N at capacity 1, the cost is not increased. We have, now, one node (i) in G_N, we start to increase the capacity (k) of our graph G_N (line 7) until all nodes belong to the graph. Whenever the capacity is increased, we are allowed to add one more node (j) and the choose the one with minimum cost.

We alternate between two steps, adding an edge node to all the nodes currently in G_N (line 11) and adding an edge to the last node (v_{k-1}) added to G_N (line 13). The algorithm can be visualized as "pushing" the minimum weight towards the right bottom side of the matrix, then the optimum for each capacity

k is set at (k, N) and the optimum for each layer i is found at $dp(i, N, N)$. Hence, the term $dp(i, k-1, N)$ can be understood as the best cost for the optimization when the capacity was $k-1$, that's the total weight we will propagate towards the end. Finally, on line 14, we take the minimum between the cost of adding the current node j and the previous cost of at the capacity k without node j.

Fig. 2 shows a complete K_4 with weighted edges and it highlights the optimum solution for the MWME problem. The highlighted node ⓓ consists of the layer which yielded the optimum solution ($i = 4$). This layer is available on Table 1. On $k = 1$, dp is initialized with node ⓓ $= 0$ (since at this capacity there are no edges in the graph). When capacity is increased to 2, the node ⓐ is added yielding a cost of 4, but within the same capacity there is a lower cost (3) if node ⓒ is added instead. The final cost is displayed at cell $j = d, k = 4$. We trace back on the same row until there is a change in cost (i.e. meaning that a node has been added). Whenever a node is added, we proceed the trace from on the row with a lower capacity $(k-1, N)$ and continue tracing back until all nodes are added, the reversed node sequence obtained by tracing back is (b, a, c, d).

Theorem 6. *The time complexity of the* `n1graph` *algorithm is* $\Theta(N^3|E|)$.

Proof. The three nested loops of i, j, k yield clearly a $\Theta(N^3)$ time. Inside the j loop, two operations will alternate: (i) adding one edge to the last node of G_N and (ii) adding an edge to each node of G_N, the total operations will be: $(1, 2, 1, 4, 1, \ldots)$, which can be split into $\lfloor \frac{N-1}{2} \rfloor$ operations of type (ii) and $\lfloor \frac{N}{2} \rfloor$ operations of type (i). The complexity for those two operations is equivalent to Eq. 9, which is the number of edges in the graph, yielding in total $\Theta(N^3|E|)$. □

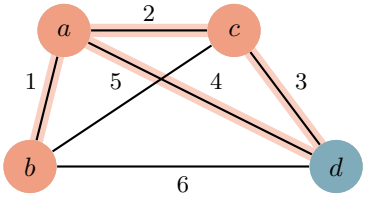


Fig. 2: A K_4 and the induced graph G_4 where ⓓ is the reference node i.

$k \diagdown j$	a	b	c	d
1	∞	∞	∞	0
2	4	4	3 ↱	← 3
3	9 ↱	← 9	← 9	← 9
4	9	10 ↱	← 10	← 10

Table 1: The trace back starts at (d,4) and moves on the same row until a value changes: a node is added to the graph.

The TraceBack starts at the right bottom corner (row N), where the minimum is, and goes back on the same line until the cost is changed. This change in cost means that a node was added (cell highlighted in orange). Remember the row means the capacity, when a node of capacity k was added, we only need to go to row $N-1$ and search for a change of node until we reach the first row. The algorithm produce a sequence in which the nodes were added, e.g. (b,a,c,d) for the Table 1. This sequence is used (as in Lemma 1) to produce G_4.

The matching is performed by bringing each point-set towards this canonical representation. Given point-sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_P$, we build the NHDD graph for each one of them. The matching associates a node with degree $v \in \mathcal{X}_k$ to a node with the same degree $v \in \mathcal{X}_l$. Therefore, the matching is performed via this

canonical representation. As each point-set is optimized individually, the scale of one does not affect the scale of the other. The optimization tries to minimize the distances within each point-set individually, therefore, it is able to handle the following two scenarios:

- **Rigid**: The sets are related by a translation and rotation: $\mathcal{X}_k = \mathbf{R}\mathcal{X}_l + \mathbf{t}$. In a rigid transformation, only the length and area are preserved.
- **Similarity**: $\mathcal{X}_l$ and $\mathcal{X}_k$ are not only related by $\mathbf{t}$ and $\mathbf{R}$, but also by an isotropic scaling s: $\mathcal{X}_k = s\mathbf{R}\mathcal{X}_l + \mathbf{t}$. This scale preserves the ratio of lengths.

We took images from the Caltech-256 dataset which contained a single object in the scene. We sampled $N = 50$ points from the sillhouete of the object. A random similarity transformation was applied to the point-sets. Figure 3 shows an example of the matching. Not all edges were displayed to avoid cluttering the scene, but the sets are correctly matched.

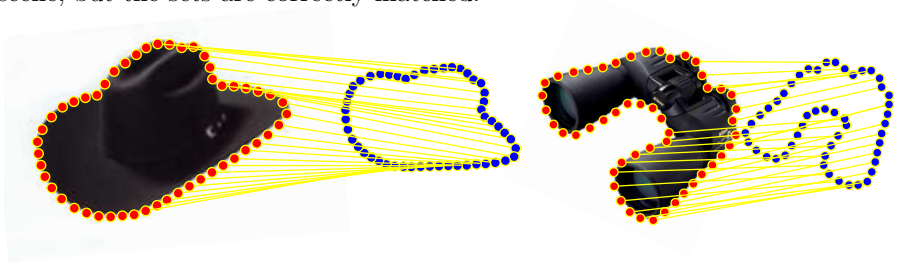


Fig. 3: Registration of point-sets under Similarity Transformation: translation + rotation + isotropic scaling.

5 Conclusions and Future Work

In this paper we proposed a novel way to match point-sets via a canonical representation in which we minimize the Min-Weight Max Entropy problem. Our approach is based on a Dynamic Programming algorithm yielding a cubic complexity solution. Such a graph turns a registration procedure into a trivial task under, for instance, rigid and similarity transformation since there is a direct mapping between nodes.

As future work, we will extend the algorithm to a more local optimization not to be dependent on all nodes. We will match regions of maximal entropy (i.e. subset of nodes). It would allow us to perform the registration of non-linearly related point-sets by performing piece-wise matchings of a subset of nodes. It will also to be able cope with noise and partially overlapping regions since it is will no longer optimize over the whole graph.

Acknowledgements

The authors would like to thank Rafael Coelho and Emir Demirovic for the discussions on the topic and Giselle Reis for the revision of the manuscript. He also acknowledges research funding from the Vienna PhD School of Informatics.

Bibliography

[1] A.D.J. Cross and E.R. Hancock. Graph matching with a dual-step em algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1236–1253, Nov 1998. ISSN 0162-8828. 1

[2] Samuel de Sousa and Walter G. Kropatsch. Graph-based point drift: Graph centrality on the registration of point-sets. *Pattern Recognition*, 48(2):368 – 379, 2015. ISSN 0031-3203. 1

[3] B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, pages 793–800, 1934. 1

[4] PeterJ. Dickinson, Horst Bunke, Arek Dadej, and Miro Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7(3): 243–254, 2004. ISSN 1433-7541. 2

[5] T. Erdős, P.; Gallai. Gráfok előírt fokszámú pontokkal. *Matematikai Lapok*, 11:264–274, 1960. 4

[6] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969. 2

[7] S. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962. 4

[8] Václav Havel. Poznámka o existenci konečných grafů. *Časopis pro pěstování matematiky*, 080(4):477–480, 1955. 4

[9] W. Kocay and D.L. Kreher. *Graphs, Algorithms, and Optimization*. Discrete Mathematics and Its Applications. Taylor & Francis, 2004. ISBN 9780203489055. 4

[10] Wei Lian and Lei Zhang. Rotation invariant non-rigid shape matching in cluttered scenes. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 506–518. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15554-3. 2

[11] B. Luo and E.R. Hancock. Iterative procrustes alignment with the {EM} algorithm. *Image and Vision Computing*, 20(56):377 – 396, 2002. ISSN 0262-8856. 1

[12] S. Ohrhallinger and S. Mudur. An efficient algorithm for determining an aesthetic shape connecting unorganized 2d points. *Computer Graphics Forum*, 32(8):72–88, 2013. ISSN 1467-8659. 2

[13] G. Reeb. Sur les points singuliers d'une forme de pfaff complétement intégrable ou d'une fonction numérique. *C. R. Acad. Sci. Paris*, 222:847–849, 1946. 2